End-to-End Packet-Scheduling in Wireless Ad-hoc Networks

V.S. Anil Kumar [1] Madhav V. Marathe [1] Srinivasan Parthasarathy [2] Aravind Srinivasan [3]

Abstract

Packet-scheduling is a particular challenge in wireless networks due to interference from nearby transmissions. A *distance-2 interference model* serves as a useful abstraction here, and we study packet routing and scheduling under this model. The main focus of our work is the development of fully-distributed (decentralized) protocols. We present polylogarithmic/constant factor approximation algorithms for various families of disk graphs (which capture the geometric nature of wireless-signal propagation), as well as near-optimal approximation algorithms for general graphs. The packet-scheduling work by Leighton, Maggs and Rao (*Combinatorica*, 1994) and a basic distributed coloring procedure, originally due to Luby (*J. Computer and System Sciences*, 1993), underlie many of our algorithms. Experimental work of Finocchi, Panconesi, and Silvestri (SODA 2002) showed that a natural modification of Luby's algorithm leads to improved performance, and a rigorous explanation of this was left as an open question; we prove that the modified algorithm is provably better in the worst-case. Finally, using simulations, we study the impact of the routing strategy and the choice of parameters on the performance of our distributed algorithm for unit disk graphs.

1 Introduction

Packet routing and scheduling are two key problems that arise in the control and design of packet-switched networks. To send a packet from a node u to node v in a network, one needs to choose a path from u to v; once the paths for all the packets have been determined, we are left with the issue of scheduling the packets along the paths. If multiple packets reach some node simultaneously, they must be queued or dropped. At most a given number of packets can be sent at a time on an edge (and sometimes nearby edges cannot have traffic simultaneously), and the scheduling problem is to decide which of the packets queued at a node should be sent first. The exact algorithms used for these problems have a significant impact on network performance. In this paper, we study the scheduling problem in the context of wireless ad-hoc networks. An *ad-hoc network* consists of a group of *transceivers* (also known as *stations*) sharing a common wireless channel and communicating with each other using this channel. Our two foci here are dealing with *interference*, and the development of *distributed* algorithms that have provably good approximation guarantees. Concretely, we are given a graph $G = (V, E)$, with packets that need to be sent from some sources to some destinations. We will also assume that paths are given; for the scheduling algorithms that we use, the algorithm of [29] can be modified to obtain good paths. If we find paths using a variant of the algorithm of [29] and then run our scheduling algorithms, we only lose an additional constant factor in the approximation ratio. We also discuss, via empirical analysis, the influence of the routing strategy on the overall performance of the protocols.

The key issue in our case is *interference*. Almost all the theoretical, algorithmic work on packet routing and scheduling so far (e.g., [14, 16]) has primarily considered the following constraint: at most one packet can be sent on an edge at a time. While useful in wired networks, this assumption is not sufficient for wireless networks, where transmission by one transceiver precludes transmission by all nearby transceivers. An example of this is shown in Figure 1: if the transmissions on (a, b) and (c, d) occur simultaneously, node b receives signals from both a and c simultaneously, which interfere with each other and get garbled; nodes a and c are not aware of this problem at b (unlike, e.g., in Ethernet, where a collision is immediately detected by all nodes). This is called the *hidden node problem* in wireless networks. On the other hand, the transmissions on (a, b) and (d, e) in Figure1 can happen simultaneously. To model this notion combinatorially, first we construct the *interference graph* for a set of transceivers by having a node for each radio, and an edge (u, v) if v lies in the transmission range of v (note that we are assuming equal transmission ranges here; see Section 2 for the general versions that we work with). We now require that the set of edges on which simultaneous

[1] Basic and Applied Simulation Science (CCS-5) and National Infrastructure and Analysis Center (NISAC), Los Alamos National Laboratory, MS M997, P.O. Box 1663, Los Alamos, NM 87545. Email: `{anil,marathe}@lanl.gov`. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[2] Department of Computer Science, University of Maryland, College Park, MD 20742. Most of the work was done while visiting Los Alamos National laboratory. Email: `sri@cs.umd.edu`.

[3] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported in part by NSF Award CCR-0208005. Email: `srin@cs.umd.edu`.

transmission happens forms a *distance-2 matching* (these definitions are given in section 2); edges (a, b) and (d, e) in Figure 1 satisfy this property. This model is called the *distance-2 interference model* [7, 9]. Earlier wireless network models (e.g. [23, 25, 24, 13]) only placed vertex constraints: nodes that transmit simultaneously must form an independent set; this is clearly inadequate, from the above description.

The above requirement might seem too restrictive sometimes: for instance, in Figure 1, b could transmit to a and c could transmit to d simultaneously; this is called the *exposed node problem* in radio networks. While this is true, reliable transmission requires transmission both ways (to acknowledge receipt of packets, for instance), and then the constraint described above seems reasonable. The 802.11 protocol addresses these issues by the MACA (Multiple Access with Collision Avoidance) algorithms (see [22] for details): a sender first transmits a *Request to Send (RTS)* signal and the receiver then sends a *Clear to Send (CTS)* signal. Once the transmission starts, the receiver sends acks to the packets he receives correctly.

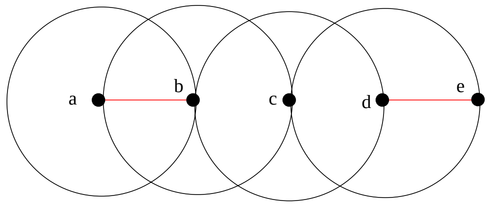


Figure 1: Distance-2 Interference

The discussion so far only addresses the restrictions that the MAC layer places in radio networks. The MAC layer only ensures delivery of packets from a node to its immediate neighbor. The *Routing Layer* of the protocol stack is responsible for choosing the routes along which packets must travel (see [22] for details), and the MAC layer moves packets one step at a time along these paths. This give us the combined routing plus packet scheduling problem: given a set of packets in a network, choose paths for the packets (routing layer function) and then schedule the packets (MAC layer function). By incorporating the above interference constraint, we have the following generalization of the problem studied in [14], which we call END-TO-END PACKET SCHEDULING WITH INTERFERENCE (EPSI): given a wireless network and a set of packets with a path given for each, develop a protocol to schedule the movement of the packets from their sources to their destinations, while ensuring that at each step packets transmitted successfully were on edges forming a distance-2 matching. Our goal is to minimize the maximum time taken by any packet to reach its destination, or the *makespan*. Thus, the EPSI problem differs from [14] in the additional constraints placed by the MAC layer.

It is not, in general, clear whether the path selection problem and the packet scheduling problems can be decoupled. In real networks, Barrett et al. [8] showed that there is significant interaction between protocols across the layers of the protocol stack. Consequently, it is not clear whether solving the corresponding problem in each layer optimally would result in optimal performance overall. The seminal paper of Leighton, Maggs and Rao [14] showed that for the standard wireline (i.e., "one packet per edge at a time" model), such a decomposition is indeed possible. Given a choice of paths for the packets, let C, the *congestion*, be defined as the maximum number of packets using any edge, and let D, the *dilation*, denote the maximum length of any path. It is easy to see that $\max\{C, D\}$ is a lower bound on the length of any schedule that uses the given paths. Leighton et al. show, quite remarkably, that the packets can be scheduled in $O(C + D)$ time. Therefore, to solve the combined routing plus scheduling problem in the wireline model, it is sufficient to obtain paths with low $C + D$ value, and then schedule the packets using [14]. Srinivasan and Teo [29] later designed an algorithm that chose paths with $C + D$ value within a constant factor of the optimal, which combined with [14] gave a constant factor algorithm for the combined routing plus scheduling problem. We remark that the problem of routing and scheduling using "standard" (i.e., distance-1 as opposed to our distance-2) matchings has been studied by [1]. Our main contributions are as follows.

(a) Packet-Scheduling Algorithms. We give the first provably-efficient distributed algorithms for packet-scheduling in networks with the distance-2 interference constraint. While the original motivation for the model is radio networks, we also study this problem on different non-geometric graph classes. As in [14], we show that the EPSI problem can be decomposed into seperate path selection and packet scheduling problems. Let n denote the number of nodes in the given network. For packet scheduling in arbitrary graphs with maximum degree Δ, we present a distributed $O(\Delta \log^2 n)$–approximation algorithm; we also show that it is hard to approximate the minimum time within a factor of $\Delta^{1-\epsilon}$ for any constant $\epsilon > 0$, even in the centralized, polynomial-time setting. As is well-known, disk graphs, defined in Section 2, model radio communication well, and we obtain the following improved approximation algorithms for packet scheduling in them. For general disk graphs, we provide a distributed $O(\log^2 n)$–approximation algorithm and a centralized $O(\log n)$–approximation algorithm. For unit-disk graphs, we give a distributed $O(\log n)$–approximation algorithm and a centralized $O(1)$–approximation algorithm. The above distributed algorithms are in the synchronous model; for unit-disk graphs, we also obtain a distributed $O(\log^3 n)$ approximation in an asynchronous model. Our results for unit disk graphs can also be extended to a more general class of graphs known as (r, s)-

civilized graphs.[4] As discussed in [13, 7], (r, s)-civilized graphs are a more realistic model for ad-hoc networks due to their ability to model occlusions, directional antennaes and varying power ranges.

A key driver of our algorithms is the *random delays* technique [14]; we combine this with additional new ideas to develop our algorithms. In particular, the standard lower bound of "congestion plus dilation" is shown to be no longer good in our model, and we use a more refined lower bound. For general disk graphs, we introduce a new packing result, which, along with the graph-theoretic notion of *inductive coloring*, helps us claim our approximation bounds. Many of our algorithms also employ a basic (list-)coloring algorithm due to [18]; we prove the improved performance of a variant of this algorithm, as discussed below.

As in [14] and other work on packet routing, our results on packet scheduling imply that the path selection problem can be decoupled from the packet scheduling problem, as long as one can find paths with low $C + D$ value, for the modified notion of congestion. For unit disk graphs, we show that the algorithm by Srinivasan and Teo [29] can be used to obtain such paths, within an $O(1)$ factor. This allows us to solve the combined routing and packet scheduling problem with only additional constant factor penalty. For arbitrary disk graphs and arbitrary graphs, unfortunately [29] only gives an $O(\Delta^2)$–approximation. But in this case, we can use the standard randomized rounding procedure due to Raghavan and Thompson [27] and get an $O(\log n)$–approximation to the path selection problem. Therefore, for the combined routing and packet scheduling problem, this leads to an additional $O(\log n)$ penalty.

(b) Distributed Coloring and Network Decomposition. Variants of the above-mentioned algorithm due to [18] have been useful in many distributed coloring algorithms (see, e.g., [12]) and also in our algorithms for packet scheduling. Due to its simplicity and generality, an extensive experimental evaluation of this and related coloring algorithms has been undertaken in [11]. A natural "non-lazy" variant of the original algorithm of [18] was empirically found to be much better. To quote [11], "In particular, when compared with Luby's algorithm it consistently turned out to be 2-3 times faster. The available asymptotic analyses do not explain this behavior ... and we leave it as an interesting open question". We present such a rigorous explanation in Lemma 7.1, showing that the non-lazy variant is provably better by a constant factor, in the worst case. In passing, we also improve the running time of a powerful distributed primitive known as *network decomposition* [4, 5, 6, 17, 21]; see, e.g., [10] for an application to distributed network protocols. We observe that the protocol of [17] which runs in $O(\log^2 n)$ time, can be modified to run in $O(\log n)$ time. In addition to being of independent interest, this is likely to be useful in a practical implementation of our algorithms.

(c) Empirical Analysis. We perform a detailed analysis of some of our algorithms, and the routing algorithm based on [3] on random geometric graphs, with a large number of packets injected at random sources. Several popular routing protocols, such as DRS and AODV [22] are based on shortest-path routing. Another popular approach is Valiant's paradigm [28]. We observe that the algorithm based on [3] yields significantly smaller congestion, compared to the other two approaches. We also study the impact of various parameters on the performance of our distributed algorithms. The two parameters of interest are the number of colors (equivalently, the frame length) used in the distributed coloring algorithm and the maximum initial random delay which each packet could be subjected to. Increasing either of the two would lead to better performance (in terms of packet losses) at the cost of increased makespan. Our studies indicate that for a wide range of values, increasing the frame length has a much bigger impact on performance rather than maximum initial random delay.

Related Work. For the earlier model of one packet per edge at a time, one of the most significant results is the work of [14], where they show the existence of a constant factor approximation, using the Local Lemma. Their result assumes that the packets already come with pre-specified paths. This work was a followed up by a series of papers (see, e.g., [15, 28, 19]. Rabani and Tardos [26] give a distributed algorithm for this problem, that takes time $O(C + D(\log^* n)^{O(\log^* n)} + \log^6 n)$, which was improved by Ostrovsky and Rabani [20] to $O(C + D + \log^{1+\epsilon} n)$. The distance-2 interference model MAC layer packet-scheduling in ad-hoc networks has been considered in [7, 23, 25]. The problem can be cast as either vertex or edge coloring depending on the particular setting [9]. The problem of finding an end-to-end schedule of packets under radio interference model has not been studied prior to this paper.

The rest of the paper is organized as follows. Section 2 defines the basic notation. Sections 3, 4 and 5 describe the packet scheduling results for disk graphs, unit-disk graphs and arbitrary graphs, respectively. The path selection is outlined in Section 6. Section 7 describes the faster distributed coloring algorithm and Section 8 discusses network decomposition. Our empirical results are described in Section 9. Many proofs and details are omitted here due to lack of space.

2 Preliminaries

Formally, an instance of the packet scheduling problem is specified as $EPSI(G(V, E), \{p_1, \ldots, p_k\})$. $G(V, E)$ is

[4]For each fixed pair of real values $r > 0$ and $s > 0$, a graph G can be drawn in R^2 in an (r, s)-*civilized manner* if its vertices can be mapped to points in R^2 so that the length of each edge is at most r, the distance between any two points is at least s, and no two edges intersect (except at their endpoints).

the underlying interference graph (described below), which would be undirected for the most part, except in disk graphs. $p_1, \ldots, p_k$ are the packets to be transmitted, with packet p_i starting at s_i and destined for t_i, along the path P_i. The path P_i is encoded in packet p_i. We will assume that any packet takes one unit of time to cross a link and at any time at most one packet can cross a link. In addition, if packets are being sent simultaneously on edges $e = (u, v), e' = (u', v')$, then $d(e, e') \geq 2$ ($d(\cdot)$ is defined below). If a packet-transmission violates either of these requirements (i.e., if some other transmission is made simultaneously on an edge that is within distance less than two), then the transmission *fails* and has to be retried later. Each node/edge has a buffer in which a packet can wait till it successfully moves to the next node in its path. The objective is to construct a *schedule*, $\mathcal{S}$, that decides which packet should be sent out at a node at any time. A schedule is *valid* iff it sends all packets along their paths subject to the above re-trial requirement (in the case of failures). Let $C_{\mathcal{S}}(p_i)$ denote the (random) time at which packet p_i is delivered in schedule $\mathcal{S}$. The *Makespan* of $\mathcal{S}$, denoted by $C(\mathcal{S}) = \max_i C_{\mathcal{S}}(p_i)$ is the time taken by $\mathcal{S}$ to route all the packets, and our objective is to construct a schedule with low (expected) makespan. For the most part, we will assume a synchronous, distributed communication model; this is reasonable in our context of a primarily local-area network, since nodes can keep synchronized, e.g., by using GPS receivers. Let Δ denote the maximum degree in G. Given subsets $A, B \subset V$ in graph $G(V, E)$, the distance $d_G(A, B)$ is defined to be the minimum length of the (directed or undirected) shortest path over all pairs of vertices $a \in A, b \in B$. For edges $e = (u, v), e' = (u', v') \in E(G)$, $d_G(e, e')$ is defined as $d_G(\{u, v\}, \{u', v'\})$. For vertex $w \in V$, we will also sometimes use $d_G(w, e)$ to denote $d_G(\{w\}, \{u, v\})$. We will drop the subscript whenever it is clear. A subset $M \subset E$ is said to be a *distance-2 matching* if $d(e, e') \geq 2$ for any distinct pair $e, e' \in M$.

Disk Graphs. A disk graph is specified by a set of points V, with a disk $D(v)$ centered at each $v \in V$, with radius $r(v)$. The directed graph $G(V, E)$ induced by these disks is the following: the set of nodes is V and a (directed) edge (u, v) is present if $v \in D(u)$. The special case where all radii are equal is called a unit disk graph, and in this case, if edge $(u, v) \in E$, then $(v, u) \in E$; as a result we can view unit disk graphs as undirected. See section 3 for more details.

3 Disk graphs

A popular model for radio networks is disk graphs. The disk around a point naturally corresponds to the effective transmission range of the radio. As described in Section 2, we will think of disk graphs as being directed. Because of our communication model, which requires two way transmission, only bidirected edges can be used for transmission; so we assume that the paths $P_1, \ldots, P_k$ only use bidirected edges. The unidirected edges only contribute to the interference: therefore, if edge $e = (u, v)$ is being used at time t, no other edge $e' = (u', v')$ with $\min\{d(u', e), d(v', e), d(u, e'), d(v, e')\} \leq 1$ can be used simultaneously. Our algorithm involves choosing random delays at the first step, as in [14] and then scheduling packets at each time step by solving a coloring problem. A sequential coloring algorithm yields an $O(\log n)$ approximation to the makespan, while a distributed coloring yields an $O(\log^2 n)$ approximation.

We need some more notation for disk graphs. For edge $e = (u, v)$, define $r(e) = r(u) + r(v)$. Define $N_{\geq}(v) = \{e' \mid d(v, e') \leq 1,\ r(e') \geq r(v)\}$ and $N_{\geq}(e) = \{e' \mid d(e, e') \leq 1,\ r(e') \geq r(e)\}$. Define $C(e)$ to be the number of packets whose path uses some edge of $N_{\geq}(e)$, and $C = \max_e C(e)$. D is still defined to be the dilation.

LEMMA 3.1. *For any vertex v, the size of the largest distance-2 matching in the subgraph induced by $N_{\geq}(v)$ is $O(1)$. Therefore, $OPT = \Omega(C + D)$.*

Figure 2 contains a description of our synchronous distributed algorithm for solving the $EPSI$ problem on disk graphs. The algorithm is called **Algorithm** DISKEPS. The description of the algorithm is complicated because of its distributed nature. The underlying sequential algorithm, based on [14], is simple: first we construct an invalid schedule $\mathcal{S}'$ which does not respect the matching constraints by first giving a random delay at the origin of each packet, and then letting it "zip through", one step at a time. We then show that at each step, for each packet p_i, $O(\log n)$ packets are transmitted simultaneously on edges not within distance-2 of it. We now construct a valid schedule $\mathcal{S}$ by considering all packets at each time step T, and moving them to their next hop, by a distance-2 coloring algorithm, DISKCOL, described below. The algorithm below is a distributed algorithm, based on [18]. Lemma 3.2 proves that this algorithm runs for $O(\log n)$ rounds and uses $O(\log n)$ colors for coloring the set E_T of edges in the algorithm DISKEPS. The lemma also proves that the additional $\log n$ factor is not needed for the sequential greedy algorithm. The algorithm's performance can be shown in the following three steps.

LEMMA 3.2. 1. *At any T (which is a multiple of $c \log^2 n$), for any edge e, define $C_T(e) = N_{\geq}(e) \cap E_T$. Then, $C_T(e) = O(\log n)$, for each e, T, with high probability.*

2. *Let T be any multiple of $c \log^2 n$. The algorithm* DISKCOL *colors the edges in E_T (defined in algorithm* DISKEPS*) in $c' \log n$ steps, using $c'' \log^2 n$ colors, with high probability. The set E_T can be colored sequentially using $O(\log n)$ colors.*

```
Algorithm DISKEPS
  1. Each packet $p_i$ chooses a delay $Z_i$ uniformly at random from $\{1,\ldots,cX_0\}$ ($c>0$ is a specific constant and $X_0 = C + D$).
     and waits for $Y_i = Z_i c\log^2 n$ steps at $s_i$.
  2. From time $Y_i + 1$ onwards, the packet moves along path $P_i$. At each node $v$ on $P_i$, the packet spends $c\log^2 n$ steps.
  3. $p_i$ reaches the $j$th node $v$ on $P_i$ by time $Y_i + j\cdot c\log^2 n$. If it reaches before $Y_i + j\cdot c\log^2 n$, it waits in the queue till then.
  4. Let $T$ be a multiple of $c\log^2 n$. All packets are moved to their next hop, from the current location at time $T$, during the time
     interval $[T,\ldots,T+c\log^2 n]$ by the following steps.
      (a) Let $E_T$ be the edges on which packets need to be moved at step $T$. Run Subroutines DISKCOL below in $c'\log n$ steps
          to choose a color $\alpha_i \in \{1,\ldots,c''\log^2 n\}$ for each packet $p_i$(this can be done, by Lemma 3.2)
      (b) At step $T + c'\log n + \alpha_i$, packet $p_i$ is moved to the next node on its path. Once it reaches $t_i$, the packet is removed.

Subroutines DISKCOL
  1. Repeat the following steps in rounds $i = 1,2,\ldots$ till all edges are colored.
  2. Round $i$: Each uncolored edge chooses to do nothing with probability $1/2$. With the remaining probability, it performs the
     following steps.
      (a) Each uncolored edge $e$ chooses a color uniformly at random from $\{(i-1)d\log n+1,\ldots,id\log n\}$.
      (b) Each edge $e$ checks whether some edge $e'$ in $N_{\geq}(e)$ has chosen the same color as $e$.
      (c) If there is no such edge $e'$, $e$ is colored with the color it chose; otherwise, $e$ remains uncolored in this round.
```

Figure 2: **Distributed algorithm for solving $EPSI$ problem on disk graphs.**

3. *Algorithm* DISKEPS *schedules the packets in time $O(\log^2 n)$ times the optimal schedule, with high probability. The sequential version of the algorithm has an approximation guarantee of $O(\log n)$.*

Proof Sketch: (1) Let each packet p_i be at the end point of edge e_i at step T. During the time interval $[T, T + c\log^2 n)$, packet $p_i, \forall i$ only moves along e_i, in an interference free manner. Now, just consider the time steps $jc\log^2 n, j = 0,1,\ldots$: with respect to these, packets wait for a random delay, and then move to their destination, one step at a time. As in [14], the expected value of $C_T(e) \leq 1$, for any such $T = jc\log^2 n$ and for any edge e. The statement now follows by a Chernoff bound.
(2) Order the edges in E_T in nonincreasing order of their $r(e)$ value. Since $C_T(e) = O(\log n)$, list coloring uses $O(\log n)$ colors for this order. For the distributed algorithm DISKCOL, in each round, each uncolored edge gets a color that does not conflict with any edges with higher $r()$ value, with constant probability. As a result, each edge gets a color in $O(\log n)$ steps, with high probability. Since $O(\log n)$ colors are used in each round, this gives a total of $O(\log^2 n)$ colors in all.
(3) The previous statements imply that each packet p_i moves one edge forward on P_i after the $Y_i c\log^2 n$ steps, with high probability for the distributed version. For the sequential coloring, this movement happens every $O(\log n)$ steps. ■

4 Unit-disk graphs

When all disks have the same radius, we obtain significant improvements in the approximation guarantee. By a repeated geometric decomposition, we can actually get an $O(1)$ approximation. This sort of decomposition only requires a sparsity condition, rather than geometry, and can be applied to bounded genus graphs also. We then give an $O(\log n)$ distributed algorithm by refining the analysis of the algorithm DISKEPS in Section 3. Finally, we give a distributed algorithm in the asynchronous model with a $O(\log^3 n)$ approximation guarantee.

4.1 A sequential $O(1)$–approximation algorithm The notation used here is defined in § 2. Assume the common radius to be 1. Let B be a bounding box in the plane for the points in V. If we assume that G is connected, B must have sides of length $O(n)$. Let B_k be a partition of B into smaller grid cells, each cell having dimensions $k \times k$. Let B'_k be obtained by translating the grid B_k by $k/2$ along the x and y axes. A cell in B_k will refer to one of the $k \times k$ sized pieces in it, and a point in B_k is any lattice point with integer coordinates. We denote lattice points within B_k by lower-case letters and the $k\times k$ cells within B_k by upper-case letters. For a disk S of radius 1 in the plane, let $C(S)$ be the number of paths P_i that visit some vertex $v \in V$, located within S. Define $C = \max_S\{C(S)\}$. As before, D is the length of the longest path. It is easy to see that $\max\{C, D\}$ is still a lower bound on the optimal size.

The main intuition for the partitioning algorithm is the following. After the first step (giving random delays and partitioning), both C and D become $O(\log n)$ within each time frame: C becomes $O(\log n)$ because of the random delays, while D becomes $O(\log n)$ because of the partitioning. This means that the smaller scheduling subproblem in any frame is localized to a $O(\log n) \times O(\log n)$ region of the plane. Thus, in addition to the temporal decomposition, we are able to do a *spatial decomposition* too. If we carry this process once more, we end up with scheduling problems on regions of size $O(\log \log n) \times O(\log \log n)$, and at this point, we can solve by brute force in $poly(n)$ time. The algorithm is described in Figure 3 and is called **Algorithm** UNITDISKEPS.

Subroutine PARTITION(k) in Figure 3 forms a partition of the problem into smaller subproblems, each on a grid of dimensions $2 \log k \times 2 \log k$.

LEMMA 4.1. *There exists a choice of random delays for all the packets in step 1 of subroutine* PARTITION(k) *which satisfies the following property: for any time frame T of length $\log k$, and for any lattice point p in the input to the subroutine, the number of paths visiting some vertex $u \in V$ located in $S(p)$ is $O(\log k)$, where $S(p)$ is a unit disk centered at p.*

LEMMA 4.2. *A schedule of length $O(\log \log n)$ can be constructed for the scheduling problem on a grid of size $O(\log \log n) \times O(\log \log n)$.*

COROLLARY 4.1. *A schedule for the distance-2 interference problem on unit disk graphs of length $O(1)$ times the optimal can be found in polynomial time.*

4.2 Distributed algorithms We start with the synchronous model. Algorithm DISKEPS detailed in Figure 2 for disk graphs can be modified to yield a better bound of $O(\log n)$ for the case of unit -isk graphs. Since all disks have the same radius, the notation and ordering of Section 3 is not needed. We will use the lower bounds C, D defined in the previous subsection.

The first three steps of the algorithm DISKEPS are unchanged, expect for the delays and time frames being multiples of $\log n$, instead of $\log^2 n$. In step 4 of the algorithm, instead of running algorithm DISKCOL, we actually run Luby's algorithm [18]. This takes $O(\log n)$ steps, and uses $O(\Delta)$ colors, where Δ is the max degree of the subgraph induced by E_T, which we is $O(\log n)$.

LEMMA 4.3. *There is a distributed, $O(\log n)$ approximation algorithm for the packet-scheduling problem on unit disk graphs.*

Asynchronous model The algorithm described earlier needs centralized, synchronous control, which is difficult in practice. We now describe a completely distributed, asynchronous, randomized algorithm that gives a schedule of length at most $O(\log^2 n)$ times the optimal, with high probability.

The basic idea is to combine contention resolution methods along with the random delays plus coloring techniques that have been used so far. Note that if there are C packets in the vicinity of some packet p, that are contending for a transmission slot at a time, all of these can be scheduled in $O(C \log n)$ steps with high probability. The random delays step allows us to reduce the effective congestion at every step, and after that one can perform coloring via the contention resolution. Note that we need to simulate some sort of synchronization, to ensure that the right set of packets is contending at any time, and this can easily be achieved by suitable waiting for polylogarithmic steps at regular intervals. We also need to keep track of the path encoded in the packet headers. The algorithm is described in Figure 4 and is referred to as **Algorithm** ASYNCHRONOUSUNITDISKEPS.

LEMMA 4.4. *Each packet p_i moves on its ℓth edge of its jth segment during time $(j-1)W + (\ell-1)W' \log n, \ldots, (j-1)W + \ell W' \log n - 1$ with high probability. Each packet p_i moves on its jth segment during time $(j-1)W, \ldots, jW - 1$ with high probability.*

COROLLARY 4.2. *All packets are delivered within time $O(OPT \log^3 n)$ with high probability, where OPT is the length of the optimal schedule.*

5 Arbitrary graphs

We now handle the case of general graphs; the maximum degree Δ of the given graph will play a key role now. We start by claiming the hardness of approximating EPSI:

LEMMA 5.1. *Let ϵ be an arbitrary positive constant. It is not possible to approximate the optimum makespan of every instance of $EPSI$ problem in polynomial time within a factor of $\Delta^{1-\epsilon}$, unless $P = NP$.*

We next present a distributed approximation algorithm using the the random delays approach from [14]. First, we need to define a better lower bound on the optimal makespan, since edge congestion is too weak: consider a complete graph K_n with one packet to be sent on every edge; the edge congestion and dilation are each 1, but the optimal makespan is $m = n(n-1)/2$. Define C as the maximum, over all edges $e = (u, v)$, of the number of paths that pass through u or v (or through both). Define D as usual to be the dilation, and let $X_0 = \max\{C, D\}$. Let OPT denote the number of steps in the optimum schedule. It is easily seen that $OPT \geq X_0$. Our algorithm GENERALEPS has two steps: (1). Construct an invalid schedule $\mathcal{S}'$ in the following manner: (a) For each packet, choose a random delay from $\{1, \ldots, cX_0\}$ ($c > 0$ is a specific constant). (b) Allow each packet to zip through along its path, after waiting for the random delay at the source. (2). Now that step (1) is done, Convert $\mathcal{S}'$ into a

```
Algorithm UNITDISKEPS
  1. Run subroutine PARTITION($n$) to create smaller problems on $2\log n \times 2\log n$ sized grids.
  2. For each of the subproblems on a $2\log n \times 2\log n$ sized grid, run subroutine PARTITION($2\log n$) to create smaller
     subproblems on $O(\log\log n) \times O(\log\log n)$ sized grids.
  3. Solve the scheduling problem within a $O(\log\log n) \times O(\log\log n)$ sized grid by exhaustive search (details in Lemma 4.2).
  4. Combine the schedules for all the subproblems together to form the whole schedule.

Subroutines PARTITION($k$)
INPUT A scheduling instance on a $k \times k$ region.
OUTPUT Partition this instance into smaller scheduling problems, defined on grid cells of size $2\log k \times 2\log k$, and compute an
invalid schedule for each subproblem.
  1. Construct an invalid schedule $\mathcal{S}_1(\Pi)$ in the following manner:
       (a) For each packet, choose a random delay from $\{1, \ldots, c(C+D)\}$, where $c > 0$ is a specific constant, such that Lemma
           4.1 is satisfied (the property in Lemma 4.1 can be checked in polynomial time; so this step involves choosing the random
           delays, checking the property and repeating if necessary).
       (b) Allow each packet to zip through along its path, after waiting for the random delay at the source.
  2. Partition $B$ into grids $B_{2\log k}$ and $B'_{2\log k}$.
  3. Consider successive time frames of length $\log k$.
  4. For each time frame $T$ of size $\log k$, assign each packet $p_i$ to a unique cell $Z$ in $B_{2\log k}, B'_{2\log k}$ such that the path traversed
     by $p_i$ during $T$ lies completely within $Z$; break ties arbitrarily.
  5. For each time frame $T$, for each cell $Z$ in $B_{2\log k}, B'_{2\log k}$, the problem restricted to $Z$ involves scheduling the packets
     assigned to it during $T$, along the segments of the paths within $Z$.
```

Figure 3: **Algorithm for solving $EPSI$ problem on unit disk graphs.**

valid schedule $\mathcal{S}$ as follows. Let E'_t be the set of edges on which packets are being transmitted at time t in $\mathcal{S}'$. Greedily distance-2 edge color edges in E'_t; let $E'_t(1), \ldots, E'_t(k)$ be the color classes (k is the number of colors used): each $E'_t(i)$ is a distance-2 matching. This can be done distributively. Schedule the packets using E'_t in k steps: at the ith step schedule packets in set $E'_t(i), i = 1, \ldots, k$.

The algorithm can be made distributed in the same way as algorithm DISKPS in § 3.

LEMMA 5.2. *Algorithm* GENERALEPS *produces a schedule of length $O((C + D) \cdot \Delta \cdot \log n)$ with high probability.*

6 Path Selection

We first describe the $O(\log n)$–approximation for $(C + D)$ using Raghavan-Thompson randomized rounding [27]. We formulate the path selection problem an integer program and then relax it, as in section 2.1 of [29]. There are two main differences from the program in [29]. The first is that C refers to our modified definition of congestion (for disk graphs or for arbitrary graphs). Next, for each edge $f \in E(G)$, the corresponding constraint in the case of arbitrary graphs becomes $\sum_{f' \in N_2(f)} \sum_{k=1}^{K} x^k_{f'} \leq C$, where $N_2(f)$ denotes the set of edges f' (including f) within distance 2 of f; for disk graphs $N_2(f)$ is replaced by $N_{\geq}(f)$. After "path filtering" as in [29], the rounding is done using [27], which yields an $O(\log n)$ approximation.

For unit disk graphs, [29] actually yields an $O(1)$–approximation, by formulating the LP differently, using *box congestion*. Partition the plane into boxes of size $1/2 \times 1/2$. It is easy to see that the discussion in Section 4 can be modified to work with this notion. Now, instead of choosing paths from a source vertex to a destination vertex, we will think of compressing each box into a single vertex, and will try to choose a path from the box containing the source to the path containing the destination. Once this is done, we can get a path in the original graph by choosing any arbitrary node for each box present in the path. Therefore, the formulation of [29] needs to be modified in the following manner: consider the graph by compressing boxes containing points into singe nodes. Each box is adjacent to neighboring boxes. The edge congestion is changed to vertex congestion, and instead of a constraint per edge in [29], we have the following constraint per nonempty box B: $\sum_{B' \in N(B)} \sum_{k=1}^{K} x^k_{B'} \leq C$, where $N(B)$ is the set of boxes within distance 2 of B, including B (there are at most 25 of these). Also, the dilation constraint is rewritten similarly. After the path filtering, the rounding in [29] can be still used because for each path P, the number of rows containing an entry for P is still $O(C)$. This gives the

```
Algorithm AsynchronousUnitDiskEPS
  1. Each packet $p_i$ chooses a delay uniformly at random from $\{1, \ldots, \alpha_1 X_0\}$, where $\alpha_1 > 0$ is a constant and $X_0$ is as defined
     before.
  2. For each packet $p_i$, define $P'_i$ to be a path obtained by prepending $X_i$ virtual edges to $P_i$, the original path (this is implemented
     by following self loop edges at the origin of $p_i$).
  3. Partition each $P'_i$ into segments of length $\log n$ each.
  4. Let $W = \alpha_2 \log^3 n$. During time $(j-1)W, \ldots, jW - 1$, each packet $p_i$ attempts to proceed on its $j$th segment in the
     following manner:
     (a) Let $W' = \alpha_2 \log^2 n$.
     (b) During time $(j-1)W + (\ell-1)W' \log n, \ldots, (j-1)W + \ell W' \log n - 1$, each packet will attempt to move on the
         $\ell$th edge of the current segment:
         i. $t$ denotes the current time; $t \in \{(j-1)W + (\ell-1)W' \log n, \ldots, (j-1)W + \ell W' \log n - 1\}$
         ii. If packet $p_i$ has already sent its $\ell$th edge of the $j$th segment, keep waiting till time $(j-1)W + \ell W' \log n - 1\}$.
         iii. If $p_i$ has not already succeeded in sending its $\ell$th edge of the $j$th segment, it chooses to send with probability
              $\frac{1}{\alpha_2 \log n}$.
         iv. If $p_i$ sent at time $t$, and detected a collision, retry as in the above step.
     (c) After packet $p_i$ succeeds in moving on the $\ell$th edge of current segment $j$, it just waits till time $(j-1)W + \ell W' \log n - 1$.
  5. If packet $p_i$ has already finished moving on the $j$th segment of its path $P'_i$, it just waits at current node till time $jW - 1$.
```

Figure 4: **Asynchronous distributed algorithm for solving $EPSI$ problem on unit disk graphs.**

following result.

LEMMA 6.1. *The path-selection problem with the objective being to minimize $(C + D)$, can be approximated to within $O(\log n)$ on arbitrary graphs, and to within $O(1)$ on unit-disk graphs.*

7 Distributed Vertex Coloring

Our packet scheduling algorithms can be viewed as implementing a distributed coloring algorithm within each frame. This motivates the question of efficient distributed algorithms for various coloring problems. Luby's algorithm [18] is often used in distributed coloring algorithms. One of the parameters in [18] is the *sleep probability* at each step, which needs to be at least $1/2$ in Luby's analysis. The algorithm works as follows on a given graph G. Each vertex $u \in V(G)$ is associated with a list $L(u)$ of colors; initially, $|L(u)| \geq \Delta + 1$, where Δ is the maximum degree in G. Vertices get colored using a distributed list-coloring algorithm in a synchronous round-by-round fashion (in a given round, any vertex communicates only with its neighbors). A generic round proceeds as follows.

(a) Each yet-uncolored vertex wakes up with probability w or goes to sleep with probability $1 - w$.

(b) Each vertex u which is awake, chooses a *tentative color* uniformly at random from its current list $L(u)$.

(c) Each vertex u that has some neighbor that chose the same tentative color as u, is called *unsuccessful*; all other (yet-uncolored) vertices are called successful.

(d) Each successful vertex v is permanently given its chosen tentative color c, and this color c is removed from $L(x)$ for all neighbors x of v such that $c \in L(x)$. The unsuccessful vertices proceed to the next round.

Note that once a vertex gets a permanent color, it is never considered again. It can be easily verified that $|L(u)| \geq d_u + 1$, where d_u is the degree of u in the current round. This also implies that step (b) is well defined; i.e., if u is yet-uncolored, $|L(u)| \geq 1$. It is also easy to verify that if and when the algorithm terminates, G has a valid vertex coloring. The empirical results of [11] showed that the algorithm improves by a constant factor when "$w = 1/2$" is changed to "$w = 1$". We provide a worst-case explanation:

LEMMA 7.1. *There are constants c and c' such that $0 < c < c'$, for which the following hold for all n large enough. (a) For any graph with n nodes, the coloring algorithm with $w = 1$, terminates within $c \log n$ rounds with high probability; and (b) on the complete graph K_n, the coloring algorithm with $w = 1/2$ requires at least $c' \log n$ rounds with high probability.*

8 Faster Distributed Graph Decompositions

A *λ-decomposition* of a graph $G = (V, E)$ is a partition of the vertex set into λ subsets (called *blocks*). The *diameter* of a decomposition is the least d such that any two vertices belonging to the same connected component of a block are at distance $\leq d$. In the distance computation, if we allow the paths between two vertices in a block to pass through other vertices which are in the same block, then the diameter is said to be *weak*; otherwise, the diameter is said to be the *strong*. In [17], Linial and Saks show that for any graph G, there exists a λ-decomposition of G such that the weak diameter of each block is at most d, where $\lambda, d = O(\log n)$. In addition, they provide a distributed algorithm which constructs such a decomposition and terminates in $O(\lambda d)$ time with high probability. We "simulate" all their rounds using a single round and obtain the following result.

LEMMA 8.1. *Our distributed algorithm decomposes any graph G in $O(\log n)$ time into $O(\log n)$ blocks with weak diameter $O(\log n)$.*

9 Empirical Analysis

We study two aspects of the packet scheduling problem empirically. The first set of experiments deal with the effect of the routing strategy on the congestion. The routing strategies we experimented with are: (i) Shortest-path routing, (ii) Valiant's paradigm [28], and (iii) Modified Source Routing (SR). The last strategy is an adaptation of the Source Routing algorithm proposed in [3]. Routes are chosen sequentially for each packet using a weighted shortest path algorithm. After each path is chosen, the weights for the edges along this path (and for some of edges which are two hops away from the path) are increased by a multiplicative factor (1.25 in our experiments). This ensures that no edges/regions in the network get overloaded by packets.

The second set of experiments deal with the sensitivity of our algorithms to various parameters: (i) the maximum initial random delay for any packet (mrd), (ii) the maximum number of colors available for the distributed edge coloring algorithm (mc), and (iii) the maximum number of rounds in the distributed coloring algorithm (nt). During a particular stage of the distributed algorithm, if a packet needs to be transmitted on some edge, and if this edge cannot be colored after nt rounds of the coloring algorithm, then the packet gets dropped. We study the sensitivity of packet loss to mrd and mc. The routing is done by SR, with $nt = 15$. The number of packets injected into the network was varied from 156 to 10000. All our experiments were performed on random connected unit disk graphs obtained by a random placement of 10000 nodes in a 50×50 square. The source and destination for each packet was chosen randomly from all nodes. The plots are obtained by averaging over ten runs.

Impact of Routing. Figures 5 (a) and (b) show the congestion and dilation in the network with respect to the number of packets in the network for the three routing algorithms. Recall that dilation is the maximum length of a path traversed by any packet. We measure congestion as follows: partition the plane into square grids of unit length; let the congestion for a specific grid be the total number of packets which pass through any node within the grid; the maximum congestion over all grids is the congestion value we plot. We use this modified definition for ease of implementation. It is easy to show that, for unit disk graphs, the modified congestion is at most within a constant factor from the previously defined congestion. We now describe the results of our experiments.

Source Routing, which is specifically tailored for reducing congestion, performs far better than either of the two strategies. Interestingly, Valiant's paradigm which is a provably good routing algorithm for hypercube networks, incurs about twice the congestion as shortest paths. We believe, this due the fact sources and destinations are chosen random from all nodes in our experiments. Hence, the advantage of choosing a random intermediate node (which may reduce congestion for fixed source-destination pairs), is not applicable any longer. Notice how the dilation varies for each strategy. After a certain number of packets, the dilation stabilizes at a constant value for all three routing algorithms. Naturally, shortest path has the least dilation of the three. Valiant's algorithm has about twice the dilation of shortest paths. This is along expected lines, since the since packets go through a random intermediate node. Source Routing has the maximum dilation of the three. While trying to route around the heavily congested regions in the network, Source Routing incurs additional costs in terms of the dilation. However, this cost is amply compensated for by the much bigger gains incurred by Source Routing with respect to congestion.

Sensitivity to Parameters. Recall that mrd and mc are the maximum initial random delay and the maximum number of colors used during distributed coloring respectively. Figures 5 (c) and (d) show the effect of mrd and mc on the total number of packets lost. For a fixed value of mc, packet loss decreases linearly with increasing values of mrd. On the other hand, for a fixed value of mrd, packet loss seems to decrease exponentially with increasing values of mc. In particular, for the range of values plotted here, doubling the value of mc yields a substantial reduction in packet loss than doubling the value of mrd. This has useful implications in practice. Increasing either of the two parameters increases the latency incurred by packets. However, in order to reduce packet loss, for a large range of values of mc and mrd, it is better to increase mc than mrd.

Acknowledgments. We thank Hari Balakrishnan, Subhash Khot and Alessandro Panconesi for helpful discussions.

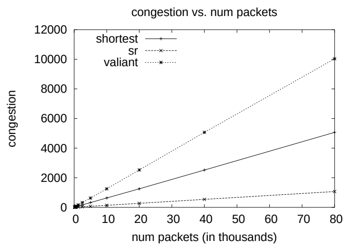
References

[1] N. Alon, F. Chung and R. Graham. Routing permutations on graphs via Matchings. *SIAM Journal of Discrete Maththematics*, 7, pp. 513-530, 1994.

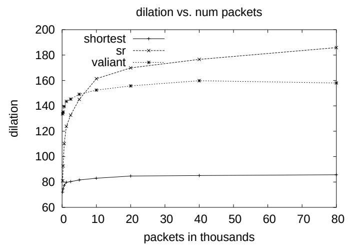
[2] M. Adler and C. Sheideler. Efficient Communication Strategies for Ad-hoc Networks. *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pp. 259-268, 1998.

[3] M. Andrews, A. Fernandez, A. Goel and L. Zhang. Source Routing and Scheduling in Packet Networks. *Proc. IEEE Symposium on Foundations of Computer Science*, 2001.

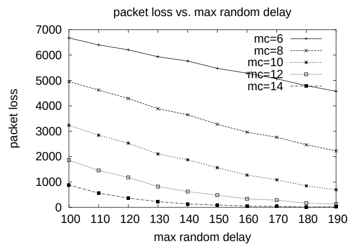
[4] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Low-diameter graph decomposition is in NC. *Random Structures & Algorithms*, 5:441–452, 1994.



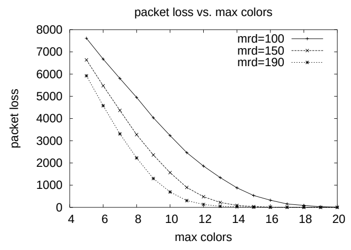
(a)



(b)



(c)



(d)

Figure 5: (a) and (b): Congestion and Dilation for different routing strategies. (c) and (d): Packet loss vs. mrd and mc

[5] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 364–369, 1989.

[6] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM J. on Discrete Mathematics*, 5:151–162, 1992.

[7] H. Balakrishnan, C. Barrett, V.S. Anil Kumar, M. Marathe, S. Thite. Induced Matchings and its Relationship to Maximum Instantaneous Capacity of Media Access Layer, Manuscript.

[8] C. L. Barrett, M. Drozda, A. Marathe and M. V. Marathe. Characterizing the Interaction Between Routing and MAC Protocols in Ad-Hoc Networks. *Proc. The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002)*, June 2002.

[9] C. Barrett, G. Istrate, V. S. Anil Kumar, M. Marathe and S. Thite. Algorithms for link scheduling in wireless radio networks. Manuscript.

[10] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 717–724, 2003.

[11] I. Finocchi, A. Panconesi and R. Silvestri. Experimental analysis of simple, distributed vertex coloring algorithms. *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pp. 606–615, 2002.

[12] D. A. Grable and A. Panconesi. Fast distributed algorithms for Brooks-Vizing colorings. *J. Algorithms*, 37:85–120, 2000.

[13] S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, 7:575–584, 2001.

[14] T. Leighton, B. Maggs and S. Rao. Packet Routing and Job Shop Scheduling in O(Congestion+Dilation) Steps. *Combinatorica*, v14, 2, pp. 167-180, 1994.

[15] T. Leighton, B. Maggs and A. Richa. Fast algorithms for finding O(Congestion+Dilation) packet routing schedules. *Combinatorica*, pp. 1-27, 1999.

[16] T. Leighton, B. Maggs, A. Ranade and S. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17:157–205, 1994.

[17] N. Linial and M. Saks. Low Diameter Graph Decompositions. *Combinatorica*, 13:441–454, 1993.

[18] M. Luby. Removing randomness in parallel computation without a processor penalty. *JCSS*, 47(2) 1993.

[19] F. Meyer auf der Heide and B. Vöcking. A packet routing protocols for arbitrary networks. *LNCS*, Vol. 439, 291-302, 1995.

[20] R. Ostrovsky and Y. Rabani. Universal O(congestion+dilation+$\log^{1+\epsilon} N$) local control packet switching algorithms. *Proc. 29th Symposium on the Theory of Computation*, pp. 644-653, 1997.

[21] A. Panconesi and A. Srinivasan. On the Complexity of Distributed Network Decomposition. *J. Algorithms*, 20:356-374, 1996.

[22] L. Peterson and B. Davie. Computer Networks: a systems approach, Third Edition. Morgan Kaufman Publishers, 1993.

[23] S. Ramanathan. Scheduling Algorithms for Multi-Hop Radio Networks. Ph.D. thesis, Department of Computer Science, University of Delaware, Newark, DE, 1993.

[24] S. Ramanathan. A Unified Framework and Algorithm for (T/F/C) DMA Channel Assignment in Wireless Networks, in *Proc. IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[25] S. Ramanathan and E. Lloyd. Scheduling algorithms for multi-hop radio networks. *IEEE/ACM Transactions on Networking*, 1:166-172, 1993.

[26] Y. Rabani and É. Tardos. Distributed packet switching in arbitrary networks. *28th ACM Symposium on the Theory of Computing*, 366-375, 1996.

[27] P. Raghavan and S. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[28] C. Scheideler. Universal routing strategies for interconnection networks, *Lecture Notes in Computer Science*, v 1390, Springer Verlag.

[29] A. Srinivasan and C-P. Teo. A constant factor approximation algorithm for packet routing, and balancing local vs. global criteria. *Proceedings of the ACM Symposium on the Theory of Computing*, pp. 636-643, 1997.